Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 36 (2014) 541 – 548
Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014-Philadelphia, PA

Simulated Annealing Approach to Solve Nonogram Puzzles with Multiple Solutions

Wen-Li Wang and Mei-Huei Tang*

School of Engineering, Penn State University, The Behrend College, Erie, PA, 16563, USA
Department of Computer and Information Science, Gannon University, Erie, PA, 16541, USA

Abstract

Nonogram, a popular Japanese puzzle game, is a well-known NP-complete problem. A number of approaches have been proposed and some algorithms are efficient in solving puzzles with one single solution. However, many puzzles are not limited to one ideal single solution. If there are multiple solutions to a puzzle, even the puzzle that is small in size may sometimes take a very long time to conquer. This type of problem is often observed to have sparse distribution of its colored cells. The existing efficient algorithms often gain no advantages, because the search space can stay huge and not reducible. For this reason, incorporating learning algorithms can be beneficial to support the deficiency. The objective of this study is to develop a heuristic means by the concept of simulated annealing (SA) to learn to explore different types of search spaces. Experiments are conducted to solve a number of multi-solution puzzles and the effectiveness of this approach is discussed.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
Keywords: Nonogram; Simulated Annealing; NP-Complete; Multiple Solutions

1. Introduction

Nonogram is one of the Japanese picture logic puzzle games and has gained in popularity. The puzzle is in a rectangular shape and the rules or constraints for playing the game are rather simple. On the side of each row or column, there is a list of numeric values associated with it. The length of the list will be the number of segments for that row or column, and the separate numbers in the list implies the length of individual segments. Nonogram puzzles can be either monochrome or in multiple colors, and this study focuses on those in black and white only. Thus, each numeric value in the list is the number of consecutive black cells for a corresponding segment. All segments belonging to a row or column are separated by at least one white cell in between. The transformation from a

* Corresponding author. Tel.: +1-814-898-6386; fax: +1-814-898-6125.
E-mail address: wxw18@psu.edu

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.052

pixel image into lists of numeric values is applicable to cryptography, allowing visual images to be encrypted. Figure 1 gives two examples. The left one has only one solution, but the other can have many solutions.

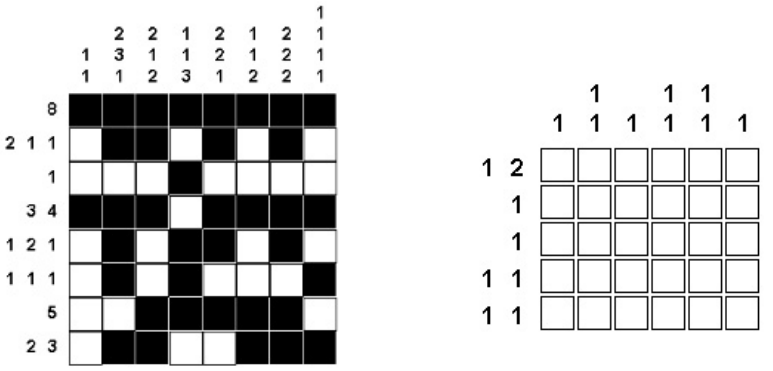

Figure 1: An example of a Nonogram puzzle

For complexity, Nonogram is similar to the complexity of coloring and scheduling problems [1, 2]. It has been proven to be NP-complete [3]. Despite its simple rules, certain puzzles are very hard to be solved in polynomial time. There are some innovative algorithms [4, 5, 6] that have been developed to efficiently solve a puzzle with one single solution. The general idea is to iteratively reduce the search space by excluding the consideration of the cells that are made certain to be black or white in color. However, the typical hardness often comes from a puzzle that is huge in size, has many solutions, and has its segments distributed sparsely. These situations often will not make the search space to be reducible to a smaller scope. Consequently, there is no progress after iterative searches so that color guessing becomes necessary for the remaining undetermined cells.

Color guessing can be done in a number of ways. Exiting algorithms may combine with bifurcation theory [7] to further narrow down the search space. Otherwise, brute force approaches [8, 9], such as depth-first search (DFS) or breadth-first search (BFS), can be adopted. Heuristic approaches [10, 11], e.g., genetic algorithms, can also be employed. In comparison, brute force algorithms do not benefit from randomness, whereas heuristic approaches can suffer from undesirable repetitions on same guessing. For a puzzle with multiple solutions, it is considered solved if one of its solutions is identified. The DFS approach is more straightforward in finding a solution than the BFS paradigm and the heuristic approaches may gain similar merits like DFS due to the randomness.

In this paper, a heuristic approach is developed based on the concept of simulated annealing (SA) [12] to solve Nonogram puzzles. The SA facilitates the utilization of randomness and the adaptation of heuristics for learning. The objective of this work is to assist in the identification of one of the solutions, especially for the case where the search space becomes difficult to be further reduced. The following sections will first describe the existing efficient algorithms and detail the limitations. The next section introduces the proposed SA heuristics. The performance of the proposed learning algorithm is discussed in the empirical study section according to a number of designed puzzles. Finally, conclusions are presented.

2. Efficient Algorithms

There are two algorithms that are able to efficiently solve Nonogram puzzles with one single solution. In [5], Yen et al. proposed an approach based on the concept of intersection. An online solver [13] adopted another means by shifting segments with the similar concept to reduce search space by iterations. The online solver also has an extended version to perform guessing. Nevertheless, it still faces the challenge to yield a solution for multi-solution puzzles within a limited time frame.

Regarding the similarity between these two paradigms, both scan through all the rows and columns in turn to identify any justifiable cells. When additional cells become sure in color, their colors are set and the same scanning process is then repeated to justify more from the remaining cells. This procedure will not stop until the puzzle is solved or no more cells can be determined. Due to the resembling concept, this section will only exemplify the procedure of the intersection-based algorithm.

Given a puzzle in Figure 2.a, all cells are initially marked with a triangle to indicate that their colors are not yet understood. Figure 2.b is the first scan result of all the rows. Based on this result, the columns are then processed in a similar manner to yield the outcome of Figure 2.c. By scanning the rows the second time, the solution is found as shown in Figure 2.d. The illustration of the intersection technique will focus on the five rows in Figure 2.b as well as the five columns in Figure 2.c. For each row and column, the corresponding list of numeric values makes the number of available combinations quite different. This approach aims at finding the intersected cells that are common in color in all of the combinations, regardless of being black or white. The identified results can be safely updated to be black or white. Figure 3 shows the intersection results for the respective rows, with which the intersection

results for the columns are further yielded as shown in Figure 4. The rows and columns pointed by an arrow sign reveal that some black and white cells can be determined through intersection. All of these determined cell results are updated into the puzzle to reduce the search space for the next iteration.

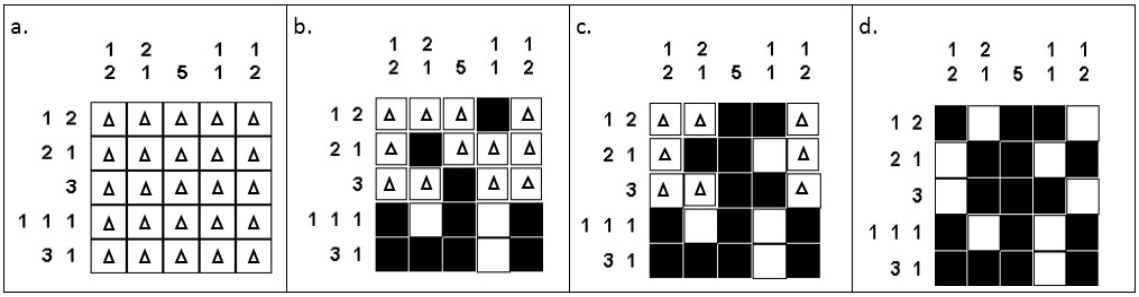

Figure 2: The procedure of intersection based algorithm

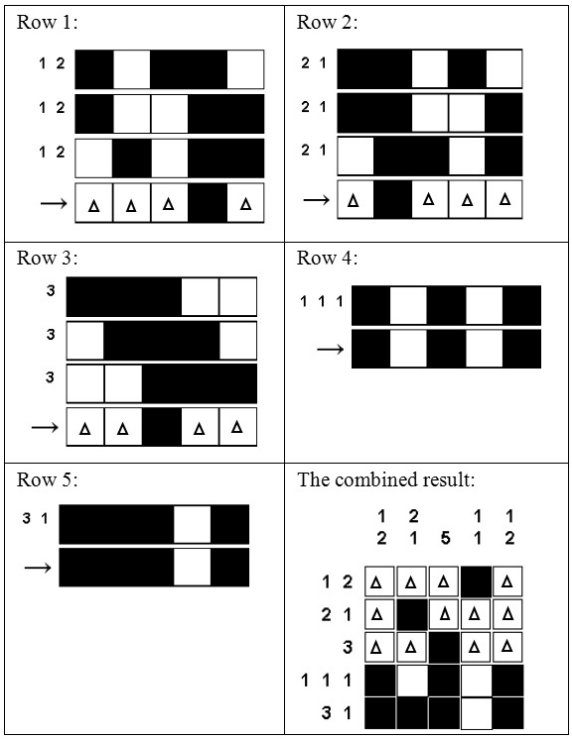

Figure 3: The intersection results of the rows of Figure 2.a.

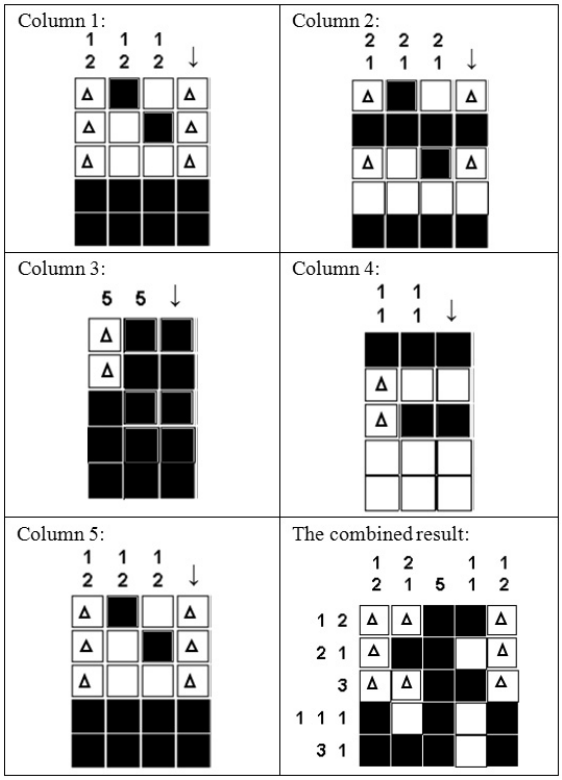

Figure 4: The intersection results of the columns of Figure 2.b.

The intersection approach achieves a good performance, because as long as there is one extra cell that can be determined in the entire rows or columns scan, the procedure makes an improvement. Therefore, for a rectangular puzzle with n rows and m columns, the puzzle can be solved in polynomial time under $n \times m$ scans. For the scan to each row and column, the algorithm can also be executed within polynomial time. Consequently, a puzzle is solvable in polynomial time.

Recall that Nonogram is a NP-complete problem. Despite the high performance of the intersection approach, it can easily fail if no more cells can be decided to be the right color in the subsequent scans. This situation often occurs to a puzzle that has many solutions, and the segments are distributed sparsely. The situation gets worse if the puzzle size is huge. Recall the sample puzzle on

the right in Figure 1. It is small in size but has multiple solutions. The intersection paradigm is however incapable of deriving any additional information to solve the puzzle. Therefore, all of the cells will remain undecided after iterations.

3. Methodology

This paper introduces a learning technique to supplement the limitations of the aforementioned scanning approaches. The technique is based on one of the optimization paradigms known as simulated annealing (SA) [12, 14]. SA borrows the concept from annealing in metallurgy where a metal material is repeatedly heated, kneaded, and cooled to enlarge the size of its crystals to eliminate defects. Fundamentally, our learning algorithm looks for conflicts to the constraints that are not yet resolved and addresses them to increasingly match with the puzzle requirements. The search behavior resembles the kneading procedure and the permission to make changes to the conflicted results corresponds to the temperature control.

The proposed SA algorithm first randomly places the total number of black cells into the puzzle, and then follows a learning approach to conduct kneading and acquire knowledge to rearrange the locations of those black and white cells to meet the rules. In regard to kneading, the learning employs heuristics to select a pair of undesirable or contradicted cells as swapping candidates. The two cells of the pair are different in color and are usually high in the degree of conflict to the solution. The heating and cooling process then dictates the learning procedure to favor or dislike the swap. In the beginning, the heat is high. The algorithm grants all sorts of swapping because it is easy to find a better solution to match with the puzzle constraints. Gradually, a better result becomes hard to be found. The temperature is thus controlled to reflect such a change and is slowly cooled down to decline unnecessary swaps. Generally speaking, it is highly desirable to admit a swap that makes an improvement to the existing result. However, there can be a dilemma. Accepting absolutely beneficial swaps that guarantee improvements may trap the result at a local optimum. Contrarily, accepting arbitrary swaps can impair the current best found solution but may somehow increase a chance to discover the global optimum. Therefore, heating and cooling should be carefully operated in order to discern a suitable circumstance for performing kneading to address the dilemma.

The development of the heuristics for computing the degree of conflict faces a challenge to conform with all the Nonogram rules simultaneously. According to the constraints, every row/column needs to have a matching number of black cells in total. These black cells should be arranged into the correct number of segments in demand. For each segment, the number of consecutive black cells, i.e., its length, must agree. The challenge comes from the difficulty of meeting both the number of segments and the segments' lengths together in learning. Before the number of segments fulfills with the demand, even though it is feasible to check the total number of black cells in a row or column, the lengths of individual segments are still not yet decidable. Therefore, the proposed heuristics below are categorized into several scenarios to determine and accumulate the severity of conflict. These scenarios are applicable to both the corresponding row and column for a cell, and the computed results are accumulated to quantify the conflict metric *c*. For generality, the following definitions and formulations utilize the term *list* to represent and model either a row or a column.

L_b^e: the existing number of black cells in the list.
L_b^r: the required number of black cells in the list.
L_w^e: the existing number of white cells in the list.
L_w^r: the required number of white cells in the list.
L_w^{2b}: the existing number of white cells that are an immediate neighbor to two black cells in the list.
L_s^e: the existing number of segments in the list.
L_s^r: the required number of segments in the list.
S_i^e: the existing number of black cells of segment *i*.
S_i^r: the required number of black cells of segment *i*.

Scenario 1: Computing the conflict metric for a cell on a list.

There are three sub-scenarios. The cell may be black or white and the total number of black cells can be a match, too abundant or insufficient. The result will be one of the followings.

- Cell is black and there are too many blacks in the list. All black cells share the responsibility to possibly turn to white. The conflict metric for the cell is computed as

$$c = \frac{L_b^e - L_b^r}{L_b^e} \quad (1)$$

- Cell is white and there are not enough blacks in the list. All white cells share the responsibility to possibly turn to black. The conflict metric for the cell is computed as

$$c = \frac{L_b^r - L_b^e}{L_w^e} \quad (2)$$

- Otherwise, there is no effect, because the existing cell color harmonizes with the existing number of black cells. Thus, the conflict metric for the cell is set to be

$$c = 0 \tag{3}$$

Scenario 2: Accumulating the conflict value for a cell on a list that has an unmatched number of segments.

There are four sub-scenarios. The cell may be black or white and the total number of *n* segments can be too abundant or insufficient. The result can be one of the followings.

- Cell is black and on the boundary of segment *i*, and there are too many segments in the list. Since turning a boundary black cell to white may reduce the number of segments, this black cell together with all other boundary black cells share the burden to change color. Note that a segment with a length one has the same cell on both sides of the boundary. Thus, the conflict metric for the cell is accumulated as

$$c+= \frac{L_s^e - L_s^r}{2n}, \text{ if } S_i^e > 1 \tag{4}$$

or

$$c+= \frac{L_s^e - L_s^r}{n}, \text{ if } S_i^e = 1 \tag{5}$$

- Cell is black and in the midst of a segment, and there are insufficient *n* segments in the list. Because turning one such middle black cell to white can increase the number of segments. Therefore, all non-boundary black cells share the responsibility for color change. Thus, the conflict metric for the cell is accumulated as

$$c+= \sum_{k=1}^{n} \left(\frac{L_s^r - L_s^e}{S_k^e - 2} \right), \text{ for each } S_k^e \geq 3 \tag{6}$$

- Cell is white and adjacent to two black cells, and there are too many segments in the list. Since changing such a white cell to black will reduce the number of segments by one, all of this type of white cells in the list share the responsibility for color change. Therefore, the conflict metric for the cell is accumulated to be

$$c+= \frac{L_s^e - L_s^r}{L_w^{2b}} \tag{7}$$

- Otherwise, there is no effect.

Scenario 3: Accumulating the conflict value for a cell on a list that has a matched number of segments.

There are three sub-scenarios. The cell may be black or white and the total number of cells in a segment can be too abundant or insufficient. The result can be one of the followings.

Cell is black and on the boundary of segment *i*, and the segment length is too long. Since turning a boundary black cell to white can reduce the length without altering the number of segments, this black cell and the other boundary black cell of segment *i* share the responsibility for color change to shorten the length. Thus, the conflict metric of this cell is accumulated to be

$$c+= \frac{S_i^e - S_i^r}{2} \tag{8}$$

Cell is white and adjacent to boundary of segment *i*, and the segment length is too short. Since changing this white cell to black can increase the length of the segment, this white cell and another white adjacent to the other boundary of segment *i* share the responsibility for color change color to increase the length. Thus, the conflict metric of this cell is accumulated to be

$$c+= \frac{S_i^r - S_i^e}{2} \tag{9}$$

- Otherwise, there is no effect.

Figure 5: The solution on top vs. the existing result at the bottom

Figure 5 gives an example for illustration. Let the top list be the solution and the list below be the current learning result. The actual solution has four segments in comparison with five segments in the current learning result. The values of those defined variables are shown in Table 1. Based on these values, the conflict metric for each cell can be evaluated. Following the scenarios of the heuristics, the stepwise computation results for all the sixteen cells in the bottom list are shown in Table 2. The *c* in total only represents a row or a column, so that many cells may get the same results. For Nonogram puzzles, the metric computation needs to account for both the corresponding row and column for each cell. This will help distinguish the cells for selection and learning. For example, cells 3, 6 and 10 have the highest value 0.5. Cell 10 is currently black while the other two are white. There are two possible pairs for swapping, i.e., (3,10) and (6,10). It can be seen that if cells 3 and 10 are swapped, the outcome becomes pretty close to the actual solution. In other words, this pair of choice yields a good learning result.

Table 1: The values for the defined variables based on the existing learning result in Figure 5

L_b^r	L_b^e	L_w^r	L_w^e	L_w^{2b}	L_s^r	L_s^e	S_i^r	S_i^e
10	9	6	7	2	4	5	4,2,1,3, respectively	1,2,3,1,2, respectively

Table 2: The computed conflicts for all the cells in the existing result of Figure 5

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Scenario 1	0.17	0	0.17	0	0	0.17	0	0	0	0.17	0	0.17	0.17	0	0	0.17
Scenario 2	0	0.20	0.33	0.10	0.10	0.33	0.10	0	0.10	0.33	0.20	0	0	0.10	0.10	0
Scenario 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c in total	0.17	0.20	0.50	0.10	0.10	0.50	0.10	0	0.10	0.50	0.20	0.17	0.17	0.10	0.10	0.17

The SA learning algorithm exploits the computed metrics to make decisions for the next move. By the idea of greed, a high percentage of kneading or swapping activities favors the pair of white and black cells with highest conflicted values. This type of swap is usually effective but the algorithm can iterate in loops without making a progress. Thus, a small percentage of learning will pick cells based on the probabilities derived from the metric values. The probability's random characteristics help escape a stoppage. For an $n \times m$ puzzle, the selection of two cell entries $c1$ in black and $c2$ in white for swapping is formulated as below.

$$c1 = \begin{cases} \text{argmax}_{e_{ij}}\{c(e_{ij})\} & \text{, if } q_1 \le \hat{q} \\ e_{ij} & \text{, where } p(e_{ij}) = \frac{c(e_{ij})}{\sum_{x=1}^{n}\sum_{y=1}^{m} c(e_{xy})} \end{cases} \text{, for all black cells} \tag{10}$$

$$c2 = \begin{cases} \text{argmax}_{e_{ij}}\{c(e_{ij})\} & \text{, if } q_2 \le \hat{q} \\ e_{ij} & \text{, where } p(e_{ij}) = \frac{c(e_{ij})}{\sum_{x=1}^{n}\sum_{y=1}^{m} c(e_{xy})} \end{cases} \text{, for all white cells} \tag{11}$$

, where e_{ij} is the cell entry of the ith row and the jth column, q_1 and q_2 are two random values between 0 and 1, and $\hat{q}$ is a specified threshold. When $q_1 \le \hat{q}$, $c1$ is the black cell e_{ij} with the highest conflict metric value $c(e_{ij})$. Otherwise, $c1$ is computed based on the probability $p(e_{ij})$ of each black cell entry e_{ij}. The same is applicable to $c2$ but for the cells to be in color white.

The temperature control concept of SA is incorporated in our learning algorithm to facilitate the greedy concept and help steer toward a different direction after a period of time of no improvements. Fundamentally, the heat is initially set to be the conflict value of the entire puzzle. Whenever an improvement can be made by a swap, the heat cools down to become the new value. If a better solution cannot be identified, the swap is rejected and the heat grows by accounting for the deficiency of this run, computed as the power to the ratio of the current poor result versus the beginning value. For the next iteration, if the new result remains higher than the old heat, the swap is rejected and the heat continues to grow. Otherwise, the swap is accepted and the heat is set to the new value. Note that the growing heat increases the opportunity of accepting a poor swap. Although the new accepted result may not be better than the previously found best solution, it provides a chance to jump out of a local optimum. The following formulates the heat h for the (n+1)th run.

$$h_{n+1} = \begin{cases} h_n + \left(\frac{h_n}{h_0}\right)^2 & \text{, if there is no improvement} \\ \sum_{x=1}^{n}\sum_{y=1}^{m} c(e_{xy}) & \text{, otherwise} \end{cases} \tag{12}$$

4. **Experiments**

Empirical studies were conducted to evaluate the performance of the SA approach to better support the deficiency of the existing algorithms. The limitations of this learning approach are also discussed. In Figure 6, four different sizes of the Nonogram puzzle were investigated. The aforementioned intersection approach, although efficient, is unable to determine the color for any cells after iterations. Basically, these puzzles mimic the situation that the existing approach has been applied but the progress reaches a point where no further improvements can be made.

The performance of the developed algorithm is measured under this system environment. The CPU is Intel dual core i7-3770 @ 3.4 GHz. The PC has 16 Gb RAM and 64-bit Window 7 Enterprise OS installed. The program runs in Java code. The collected elapsed time for the four puzzles is the average result of hundreds of random runs and the threshold value $\hat{q}$ for Eqs. 10 and 11 is set to be 0.9. It can be seen that the time goes up exponentially with the size of the puzzle being slightly increased, as shown in Table 3.

For the size of 10 by 10, it takes only around .22 second to solve the puzzle. For the largest one of 25 by 25, it then spent more than an average of 4 minutes to yield a solution. The huge time difference is due to the number of combinations to meet the lists of constraints grows significantly huge, especially with the segments being short in length and sparsely allocated. Even though there are multiple solutions available to the puzzles, if the combinations cannot be reduced to a smaller amount, the search space will remain large and the performance is dramatically degraded.

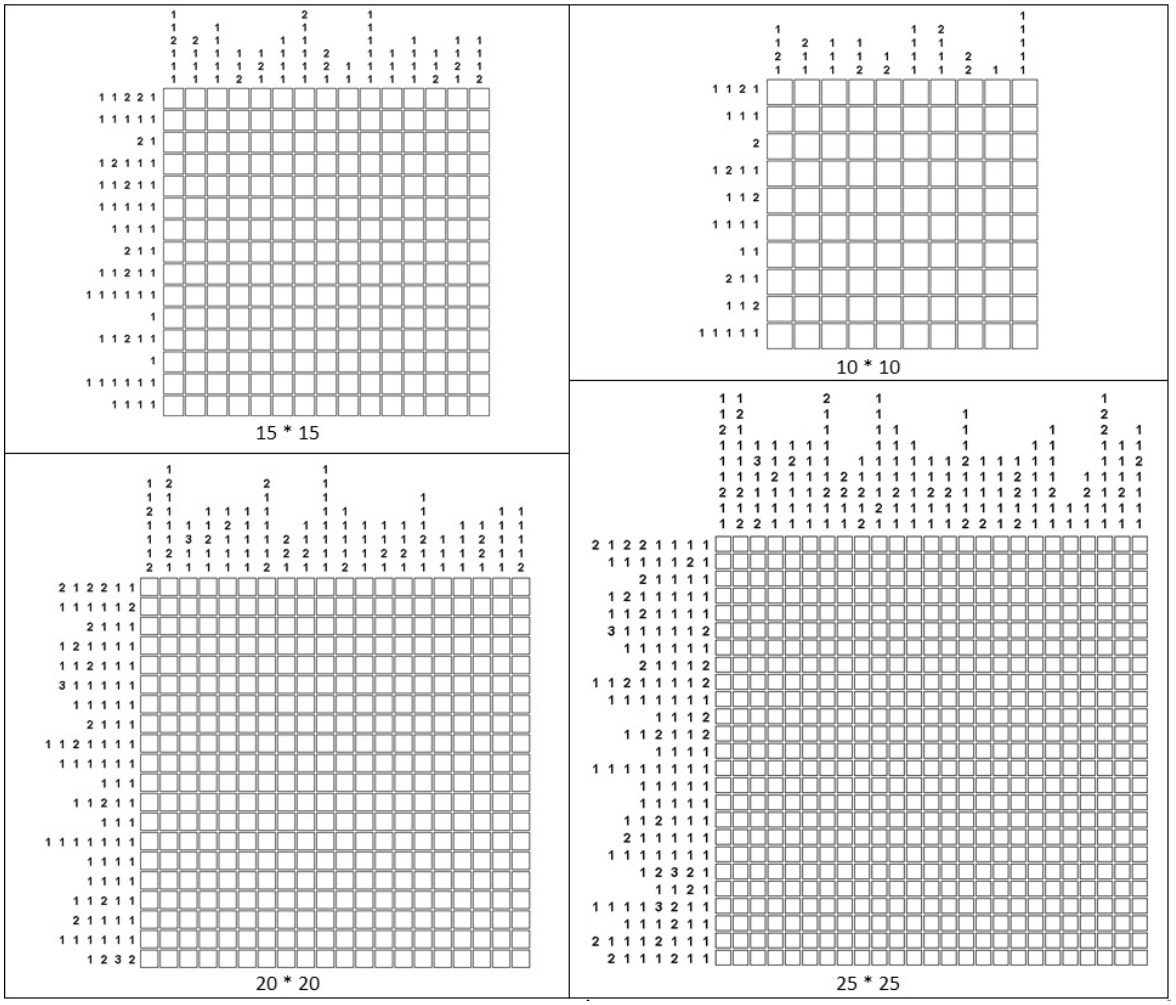

Figure 6: The performance for solving puzzles of different sizes

Table 3: The performance comparison for puzzles in different sizes

	10 * 10	15 * 15	20 * 20	25 * 25
Time	.22 second	.92 second	11.66 seconds	251.69 seconds

5. **Conclusions**

A Nonogram solver based on simulated annealing (SA) was developed to compensate the deficiency of the existing algorithms, which fell short in solving puzzles with many possible solutions. Regarding the NP-complete complexity of Nonogram, the hardness typically comes from a puzzle that is huge in size, has small segments sparsely distributed, and has many solutions. Because puzzles

with many solutions will not absolutely limit a cell to a certain color, the existing approaches that scan rows and columns repeatedly may not be able to capture new knowledge for the next iteration to follow. Under such a situation, the search space cannot be eliminated to gain performance. Our SA approach incorporates learning heuristics to help resolve the issue.

The proposed heuristics are developed to compute the conflict metric and for each heuristic a number of criteria are further categorized to guide the computations for each cell. For the kneading process of SA, our approach employs a greedy algorithm to choose highly conflicted cells to perform color swap in order to aggressively improve the solution. The approach also incorporates dynamic temperature control to prevent the greedy approach from sticking at a local optimum. The heat will gradually cool down if the conflicts to the constraints are constantly reduced. This facilitates a better solution to be found. However, the heat is raised up after a number of iterations without improvements to permit a poor swap to be accepted. In this way, the algorithm has a chance to jump out of the local optimum. In summary, the temperature control encourages better solutions to be continuously identified, and can also re-direct the search direction if a better solution is hard to be found after a certain period of trials. Experiments are conducted on four multi-solution puzzles for which the existing intersection approach cannot gain any knowledge to make improvements. The results show that our SA learning algorithm can yield a solution for all of them. Nevertheless, the solving time increases exponentially with the puzzle size.

Our future work will enhance the heuristics to handle bigger size puzzles. Another research direction is to either meet the constraints in advance for the row segments or the column segments to reduce the consideration from two dimensions to just one single dimension. This way will facilitate the movement of the entire segment instead of each individual cell.

References

[1] M. Dániel, "Graph Colouring Problems and Their Applications in Scheduling", *Periodica Polytechnica Ser. El. Eng.* vol. 48(1), 2004, pp. 11-16

[2] T. Erlebach and K. Jansen, "The Complexity of Path Coloring and Call Scheduling", *Journal of Theoretical Computer Science*, vol. 255(1-2), 2001, pp. 33-50

[3] N. Ueda; T. Nagao, "NP-completeness Results for Nonogram via Parsimonious Reductions", *TR96-0008, Technical Report*, Department of Computer Science, Tokyo Institute of Technology, May 1996

[4] K. J. Batenburga; W. A. Kosters, "Solving Nonograms by combining relaxations", *Journal of Pattern Recognition*, vol. 42(8), August 2009, pp. 1672–1683

[5] S. J. Yen; T. C. Su; S. Y. Chiu; and J. C. Chen, "Optimization of Nonogram's Solver by Using an Efficient Algorithm", *Proceedings of International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 18-20 November 2010, pp. 444–449

[6] C. H. Yu; H. L. Lee; L. H. Chen, "An Efficient Algorithm for Solving Nonograms", *Applied Intelligence*, vol. 35(1), 2011, pp. 18–31

[7] R. Rand; D. Armbruster, *Perturbation Methods, Bifurcation Theory and Computer Algebra*, Applied Mathematical Sciences Series, Springer, 4 October, 2013

[8] M. Q. Jing; C. H. Yu; H. L. Lee; L. H. Chen, "Solving Japanese Puzzles with Logical Rules and Depth First Search Algorithm", *Proceedings of the IEEE Eighth International Conference on Machine Learning and Cybernetics*, Baoding, 12-15 July 2009, pp. 2962-2967

[9] W. A. Wiggers, "A Comparison of a Genetic Algorithm and a Depth First Search Algorithm Applied to Japanese Nonograms," Twente Student Conference on IT, June 2004

[10] J. T. Tsai; P. Y. Chou; J. C. Fang, "Learning Intelligent Genetic Algorithms Using Japanese Nonograms", *IEEE Transactions on Education*, vol. 55(2), 2012

[11] S.S. Sancho; G. O-G. Emilio; M. P-B. Angel; P-F. Antonio; X. Yao, "Solving Japanese Puzzles with Heuristics", *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, Honolulu, Hawaii, 1-5 April 2007, pp. 225-231

[12] V. Granville; M. Krivánek; J.P. Rasson, "Simulated Annealing: A Proof of Convergence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16.6, 1994, pp. 652-656

[13] http://www.comp.lancs.ac.uk/~ss/nonogram/

[14] S. Kirkpatrick; C.D. Gelatt Jr.; M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220 (4598), 1983, pp. 671–680